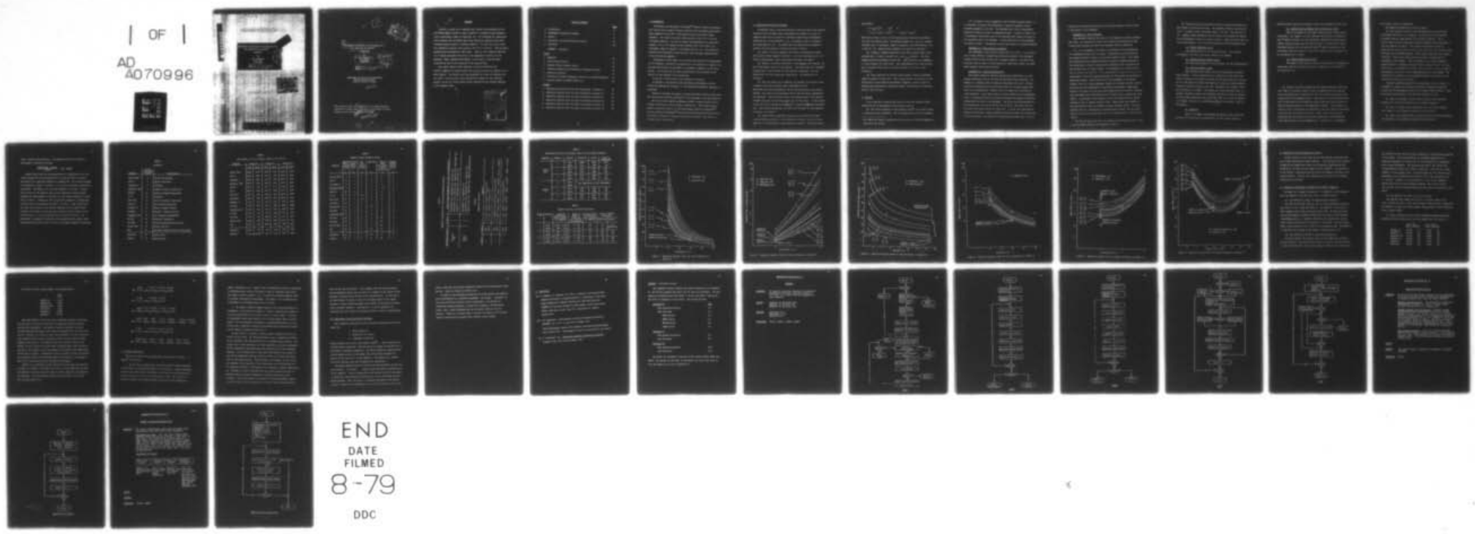
AD-A070 996 TORONTO UNIV (ONTARIO) DEPT OF ELECTRICAL ENGINEERING F/G 9/2
PERFORMANCE PREDICTION AND EVALUATION FOR THE AN/UYK-20 AND AN/--ETC(U)
1975 V C HAMACHER, S G ZAKY, S RANNEM N00014-75-C-1174

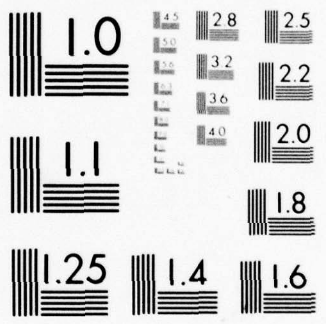
UNCLASSIFIED NL

| OF |

AD
A070996



END
DATE
FILMED
8-79
DDC


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

Performance Prediction and Evaluation for the AN/UYK-20 and AN/UYK-7 Computers

NRL

1

by

V.C. Hamacher
S.G. Zaky
S. Rannem

code (1)

(6)

Performance Prediction and Evaluation for the AN/UYK-2Ø and AN/UYK-7 Computers

by

(10) V.C. Hamacher
S.G. Zaky
S. Rannem

(11) 1975

(12) 43p.

Department of Electrical Engineering
University of Toronto
Toronto, Ontario, CANADA

1975

This research report was prepared for the Naval Research Laboratory through support provided by the Office of Naval Research under contract NØØØ14-75-C-1174

(15)

355515

1

ABSTRACT

A review is given for a technique that relates the performance of the CPU $\leftrightarrow$ Main Memory section of small computers to standard design parameters such as wordlength, number of registers, etc. The technique was constructed from execution time and memory space data obtained by applying three small benchmark program kernels to fifteen computers. The data was used to determine regression equations that provide a best fit to the data. Time and space equations were developed for each kernel. Three or four variables from the set of design parameters were used as the independent variables in each equation. These variables were chosen, in each case, as the ones that accounted for most of the variation in the observed data.

This report applies these equations to both the AN/UYK-20 and AN/UYK-7 computers to predict their time and space values with respect to the three kernels. The kernels have been programmed for these two machines and the actual time and space values obtained are compared to the predicted results, as well as to the actual results obtained for the fifteen computers in the original study.

C

TABLE OF CONTENTS

1. Introduction

The authors have developed a technique[1] based on regression equations that can be used to predict relative execution time and memory space performance measures for the CPU $\leftrightarrow$ Main Memory section of a range of small computers. The range is basically characterized as that of 8- to 24-bit wordlength computers. In order to predict the performance of a particular computer, only a few standard design parameters need to be known, e.g., wordlength, number of registers, byte-addressability, etc. The regression equations were developed by applying three small benchmark program kernels to each of fifteen computers.

The purpose of this report is to describe the results of the development of the technique, followed by an application of the equations to predict time and space performance of the AN/UYK-20 and AN/UYK-7 computers on the three kernels. The actual time and space requirements of these machines on the kernels is then compared to the predictions.

In Section 2 of the report, the technique is described.

In Section 3, the details of the technique are documented, and the results of applying the technique to the AN/UYK-20 and AN/UYK-7 computers is presented.

Section 4 discusses the general technique and offers specific conclusions that may be drawn from the application of the technique to the above machines.

We do not review computer performance methods in general and we do not attempt to justify the use of kernels in particular. These aspects can be found by consulting articles in the bibliography compiled by Agajanian[2]. The text edited by Freiberger[3] contains detailed discussions of the use of statistical techniques in computer performance evaluation. Our work is in the same spirit as this text.

2. Performance Evaluation Technique

We postulate that it should be possible to get some idea of the relative execution times and memory space requirements for members of a class of computers when they are placed in a particular application environment by examining their design parameters, e.g., wordlength, number of registers, byte-addressability, etc. The method that the authors have developed to quantify the relationship is briefly described as follows:

(a) Choose a class of computers and select a representative subset. [We selected fifteen computers from the 8- to 24-bit wordlength class; so a focus on minicomputers, and a little above and below, was taken.]

(b) Specify, at the flow-chart level, a few programs that exercise the CPU $\leftrightarrow$ Main Memory section of the machines. [We chose three small benchmark program kernels from the areas of high precision arithmetic, character manipulation, and list processing, respectively. No input/output was involved.]

(c) Code all kernels on all machines and evaluate the executior times in memory cycles and the memory space requirements in bits. [Execution times were determined by the use of an abstracted trace routine program. This tracer was constructed from the flow of control exhibited by the flowcharts. Hand calculated values for execution times of the various straight line parts of the kernels were input to the tracer. It then followed flow of control (looping and branching) to accumulate ~~the~~ total execution time. Some uniform assumptions were made about frequency of data dependent branching, as required.]

(d) Choose forms of equations (regression fit analysis) that have standard machine parameters as the independent variables and time (T) and space (S), as defined above, as the dependent variables. [The forms chosen were either:

$$Y = c_0 x_1^{c_1} x_2^{c_2} \ldots x_n^{c_n} \quad \text{or}$$

$$Y = c_0 + c_{1,1}x_1 + c_{1,2}x_1^2 + \ldots + c_{n,1}x_n + c_{n,2}x_n^2 .$$

The first of these we will call the multiplicative form and the second we will call the additive form. The machine parameter set, the x_i's, includes variables such as wordlength, number of registers, byte-addressability, etc. A total of six machine design parameters was found to be adequate.]

(e) Perform a standard regression fit of these equation forms to the observed data in order to evaluate the c_i and $c_{j,k}$ constants. The least squares criterion determines the best fit. [This results in six equations, a T and S equation for each kernel. Some experimentation was used to determine i) the best form, and ii) the three or four x_i's to use in each situation.]

(f) These equations can then be used to predict relative performance among all machines of the class. [This report presents the results of such predictions for the AN/UYK-20 and AN/UYK-7 computers, and the results of checking the predictions by coding the kernels on each machine to determine actual T and S values.]

3. Results

It is convenient to present the results in the same sequence as the description of the methodology in the previous section.

(a) The fifteen computers of the original study[1] are listed in Table 1* along with their wordlengths. The two AN/UYK machines are also included.

* All tables and figures referenced in this section are collected together at the end of the section.

(b) In order to select appropriate small benchmark program kernels, it is necessary to consider some application areas and formulate a small benchmark program in each area. The benchmarks must then be individually analyzed in order to extract an appropriate kernel. A kernel is taken to be a structurally identifiable part of the benchmark that accounts for most of the execution time of the benchmark. A brief description of the benchmarks and kernels follows. The Appendix contains more details, including flowcharts.

Benchmark #1: High Precision Arithmetic

The wordlength of the machines ranged from 8 to 24 bits. As an indication of their ability to handle high precision arithmetic, this benchmark performs 48-bit integer division by a standard technique. Over 90% of the execution time was spent in three subroutines named MULAD, MULSB, and SHFTM. These routines constituted the kernel.

Benchmark #2: Character Manipulation

The crucial problem in character manipulation applications is to use storage effectively, and at the same time facilitate fast processing. For example, in a machine where the smallest addressable data unit is 16 bits, two bytes or characters must be packed per data unit if good storage efficiency is to be maintained. However, this will impede the accessing of a single character. Since memory space is usually limited in the minicomputer class, it was decided that on all machines maximum core packing of character strings would be used in this benchmark. The actual processing problem was the construction of a file of records to be printed. This print file was extracted from certain fields of a base file. A format list specified the fields to be selected. Linear searching of both the format list and the base file were involved in the process of constructing the print file. A set of routines that accounted for about 65% of the total execution time was chosen as the kernel of this benchmark.

Benchmark #3: List Processing

This benchmark exercises the ability of the machines to handle scattered data items. An algorithm for binary tree insertion and balancing was used. A kernel that accounted for 80% of the execution time was identified.

(c) In all benchmarks, the method of proceeding to the determination of an appropriate kernel was as follows. Flow-charts were constructed for the complete benchmark without any particular machine in mind. Even though these flow-charts were reasonably detailed, there was enough flexibility at the coding phase to exploit the particular strengths of the instruction set and CPU facilities of a given machine. The benchmarks were actually run on only one machine (the PDP-8/I). This established the logical correctness and completeness of the flow-charted algorithms and facilitated the extraction of appropriate kernels. These kernels were then programmed at the machine assembly level by a single programmer, Rannem, on the other fourteen machines. Rannem was also the programmer for the AN/UYK-20 and AN/UYK-7 experiments. Because of the small size and easy understandability of the kernels, execution times and memory space requirements were relatively easy to compute for these kernels. As mentioned in the previous section, an abstracted trace routine program was used to compute execution times. Memory space was recorded in bits and execution time was recorded in number of memory cycles. The latter parameter allows a concentration on machine design features, and the actual speeds of the various technologies used in the machines have no effect on the results.

The time and space values for all machines of the original study, as well as for the AN/UYK machines are displayed in Table 2.

(d) The multiplicative and additive forms of regression equations have been stated in the previous section. The dependent variables are execution time, T, in memory cycles, and memory space, S, in bits. The final set of six computer design parameters that were found to be adequate for explaining the performance results are listed in Table 3, along with their values on each of the machines. We list brief explanations of these parameters here:

x_1: Memory Wordlength (bits)

The maximum number of bits per memory access. This normally corresponds to the basic instruction length of the computer.

x_2: Minimum Bytes per Memory Access

This identifies whether or not the machine has byte addressability.

x_3: Add Time (memory cycles)

Most machines have a number of ways of determining an effective memory address. We have standardized on a definition of add time as the number of memory cycles needed to perform the add instruction with one operand in a CPU register and the other operand in a directly addressed memory location, leaving the answer in the CPU. The memory cycle needed to fetch the instruction is included. This definition of add time means that x_3 is actually a general parameter that probably indicates the speed of execution of other binary operations such as AND, MASK, SUB, etc., where one operand is in memory, the other is in ~~a CPU~~ register, and the result is left in a CPU register.

x_4: Registers

This is the number of wordlength CPU registers that can be used generally for holding both operands/results and main memory addresses.

Dedicated index registers and program counters are excluded from this count.

x_5: Address Reach per Memory Word of Instruction (bits)

This is the number of main memory address bits per memory word of instruction. For example, a 12-bit wordlength machine that can explicitly name 2^8 memory locations in a one word memory reference instruction has an address reach per memory word of instruction of 8, (= 8/1), while a 16-bit wordlength machine that can explicitly name 2^{16} memory locations in a two word memory reference instruction also has an address reach per memory word of instruction of 8 (= 16/2).

x_6: Address Modification (bits)

The number of bits that are used in an instruction to determine the addressing mode for accessing an operand. Examples are the indirect bit, the index bit, etc.

(e) Based on the data developed for the fifteen machines in the original study, as recorded in Table 2, best fit equations for both time and space were heuristically determined for each of the three application areas represented by the benchmark kernels. This heuristic search involved trying both the multiplicative and additive equation forms with various subsets of machine design parameters as the independent variables. This process, its accuracy, and its limitations are discussed in the next section of the report. We present only the final results here. Table 4 lists the subsets of parameters that were found to be the most significant in determining the performance results in each benchmark area. In each case, the parameters are listed in order of significance.

The regression equations are listed in Table 5.

(f) The equations of Table 5 were determined from the data derived from coding the kernels on 15 computers. We can now use these equations to predict time and space measures for the AN/UYK computers. This is done by substituting the appropriate computer parameter values from Table 3 into the equations of Table 5. The results are displayed in Table 6, along with the actual values of time and space that were determined by coding the kernels on the AN/UYK computers. These actual values were also entered in Table 2. There are no values entered for kernel #1 (high precision arithmetic) on the AN/UYK-7. This is because it is a 32-bit wordlength machine and the #1 benchmark involved a 48-bit integer division. All other machines have a wordlength that divides evenly into 48, so that it would be misleading to try to fit a 32-bit wordlength machine to this program. The suitability of including a 32-bit wordlength machine in the study will be discussed in Section 4 of the report.

In order to gauge the quality of the predictions listed in Table 6, it is helpful to examine the closeness with which the regression equations actually fit the performance values of the original fifteen machines. Three questions that seem natural to ask about the original fits are:

Q1: What is the range of actual values of each of the performance parameters for each of the kernels over all fifteen computers?

Q2: What is average error in prediction over all fifteen computers with respect to each performance parameter for each kernel?

Q3: What is the range of error in prediction over all fifteen computers with respect to each performance parameter for each kernel?

Table 7 answers these questions. All prediction errors are stated as percentages, calculated as follows:

$$\frac{|\text{PREDICTION - ACTUAL}|}{\text{ACTUAL}} \times 100 = \text{\%ERROR}$$

Another useful form for the presentation of a regression fit is via a scatter diagram of the experimental data on which the family of curves generated by the regression equation is superimposed. Six of these diagrams are presented in Figures 1 through 6, one for each performance parameter for each kernel. The format of all of these diagrams is the same. A performance measure is the ordinate, and the most significant machine design parameter for that measure is the abscissa This design parameter is listed first in Table 4. Holding the other significant parameters at appropriate constant values then generates the family of curves. These values cover the range that occurred in the fifteen computers of the original study. Therefore, the extent to which the curves span the space occupied by the scatter points is a visual indication of the quality of the fit. The unlabelled X's represent the data from the original fifteen machine study. The predicted and actual values for each of the AN/UYK computers are labelled.

TABLE 1

Computers

Computer	Memory Wordlength (bits)	Manufacturer
Varian 520/i	8	Varian Data Machines
SPC-12	8	General Automation
Interdata 1	8	Interdata
Datapoint 2200	8	Computer Terminal Corporation
PDP-8/I	12	Digital Equipment Corporation
H 112	12	Honeywell
PDP-11/20	16	Digital Equipment Corporation
Supernova	16	Data General Corporation
Modcomp III	16	Modular Computer Systems
DataMate	16	DataMate Computer Systems
Kongsberg 400	16	A/S Kongsberg Vaapenfabrikk
DC 6024	24	Datacraft Corporation
SEL 804A	24	Systems Engineering Laboratories
GE-PAC 4010	24	General Electric
SAM	24	Norwegian Defence Research Establishment
AN/UYK-20	16	Sperry Univac
AN/UYK-7	32	Sperry Univac

TABLE 2

Time (memory cycles) and Space (memory bits) Results

Computer	Kernel #1		Kernel #2		Kernel #3	
	Time	Space	Time	Space	Time	Space
Varian 520/i	23866	1576	1890	1352	19633	2192
SPC-12	33462	2064	2123	1752	23043	2928
Interdata 1	41532	3016	2223	1456	50698	4248
Datapoint 2200	49345	2104	3086	1664	41294	3584
PDP-8/I	11448	1224	3627	1776	19305	2448
H 112	23475	1452	7027	1880	29767	2532
PDP-11/20	6071	944	1597	1584	10796	2088
Supernova	5460	1072	2818	2192	10000	2416
Modcomp III	4616	1120	1012	1520	7256	1904
DataMate	6296	1328	2533	1872	9440	2240
Kongsberg 400	4809	1072	2515	1872	7805	2016
DC 6024	1365	768	936	1968	5778	2472
SEL 804A	1576	936	4989	3024	8942	3048
GE-PAC 4010	2962	1248	6793	3456	12615	3144
SAM	1535	864	4640	2904	7288	2640
AN/UYK-20	5202	992	922	1040	7580	1808
AN/UYK-7	not applicable		881	2240	5421	3136

TABLE 3

Computer Design Parameter Values

Computer	Memory Word-length (bits)	Minimum bytes/ memory probe	Add Time (memory cycles)	Registers	Address Reach per memory word of instruction	Address modifi-cation (bits)
	x_1	x_2	x_3	x_4	x_5	x_6
Varian 520/i	8	1	3	7	5	3
SPC-12	8	1	5	5	6	0
Interdata L	8	1	3	1	4	1
Datapoint 2200	8	1	7	5	3.2	0
PDP-8/I	12	2	2	1	8	1
H 112	12	2	4.5	1	8	1
PDP-11/20	16	1	4.2	6	8	3
Supernova	16	2	3	4	8	3
Modcomp III	16	1	3	15	8	4
DataMate	16	2	2	2	8	3
Kongsberg 400	16	2	2	6	8	3
DC 6024	24	1	2	5	15	3
SEL 804A	24	3	2	5	15	3
GE-PAC 4010	24	3	2	1	15	4
SAM	24	3	2	10	14	4
AN/UYK-20	16	1	3	16	8	4
AN/UYK-7	32	1	2	15	16	7

TABLE 4

Important Parameters for Predicting Performance Measures

Performance Measure	Kernel #	Machine Design Parameters (in order of significance)
Execution Time	1	Wordlength (x_1), Registers (x_4), Add Time (x_3)
	2	Bytes/probe (x_2), Add Time (x_3), Registers (x_4), Wordlength (x_1)
	3	Registers (x_4), Wordlength (x_1), Add Time (x_3)
Memory Space	1	Wordlength (x_1), Registers (x_4)
	2	Bytes/probe (x_2), Wordlength (x_1), Registers (x_4)
	3	Address Reach (x_5), Registers (x_4), Address Modification (x_6)

TABLE 5

Regression Equations

Kernel #	Measure	Equation
1	Time	$T = 2.29 \times 10^6 (x_1^{-2.26})(x_4^{-0.276})(x_3^{0.640})$
	Space	$S = 1.01 \times 10^4 (x_1^{-0.720})(x_4^{-0.109})$
2	Time	$T = 1780(x_2^{1.48})(x_3^{0.687})(x_4^{-0.155})(x_1^{-0.290})$
	Space	$S = 2130 - 964x_2 + 385x_2^2 + 2.97x_1 + 0.807x_1^2 - 18.7x_4 - 0.296x_4^2$
3	Time	$T = 1.35 \times 10^5 (x_4^{-0.343})(x_1^{-0.885})(x_3^{0.497})$
	Space	$S = 6140 - 658x_5 + 33.3x_5^2 - 149x_4 + 6.83x_4^2 - 252x_6 + 41.7x_6^2$

TABLE 6

Predicted and Actual Performance Values for the AN/UYK Computers

Computer	Kernel #	Measure	Predicted	Actual	Error in Prediction
AN/UYK -20	1	Time	4088	5202	21%
		Space	1014	992	2%
	2	Time	1102	922	20%
		Space	1430	1040	37%
	3	Time	7741	7580	2%
		Space	2039	1808	13%
AN/UYK -7	1	Time	(not applicable on this benchmark)		
		Space			
	2	Time	689	881	22%
		Space	2125	2240	5%
	3	Time	3503	5421	35%
		Space	3717	3136	19%

TABLE 7

Relative Quality of AN/UYK Predictions

Kernel #	Measure	Range of Actual Values over original 15 computers	Range of Error in Prediction over original 15 computers	Average Error in Prediction over original 15 computers	Error in AN/UYK Predictions	
					AN/UYK-20	AN/UYK-7
1	Time	1,365 → 49,345	2 → 26%	10%	21%	not applicable
	Space	768 → 3,016	5 → 37%	14%	2%	
2	Time	936 → 7,027	0 → 35%	11%	20%	22%
	Space	1,352 → 3,456	0 → 13%	6%	37%	5%
3	Time	5,778 → 50,698	4 → 29%	20%	2%	35%
	Space	1,904 → 4,248	2 → 18%	8%	13%	19%

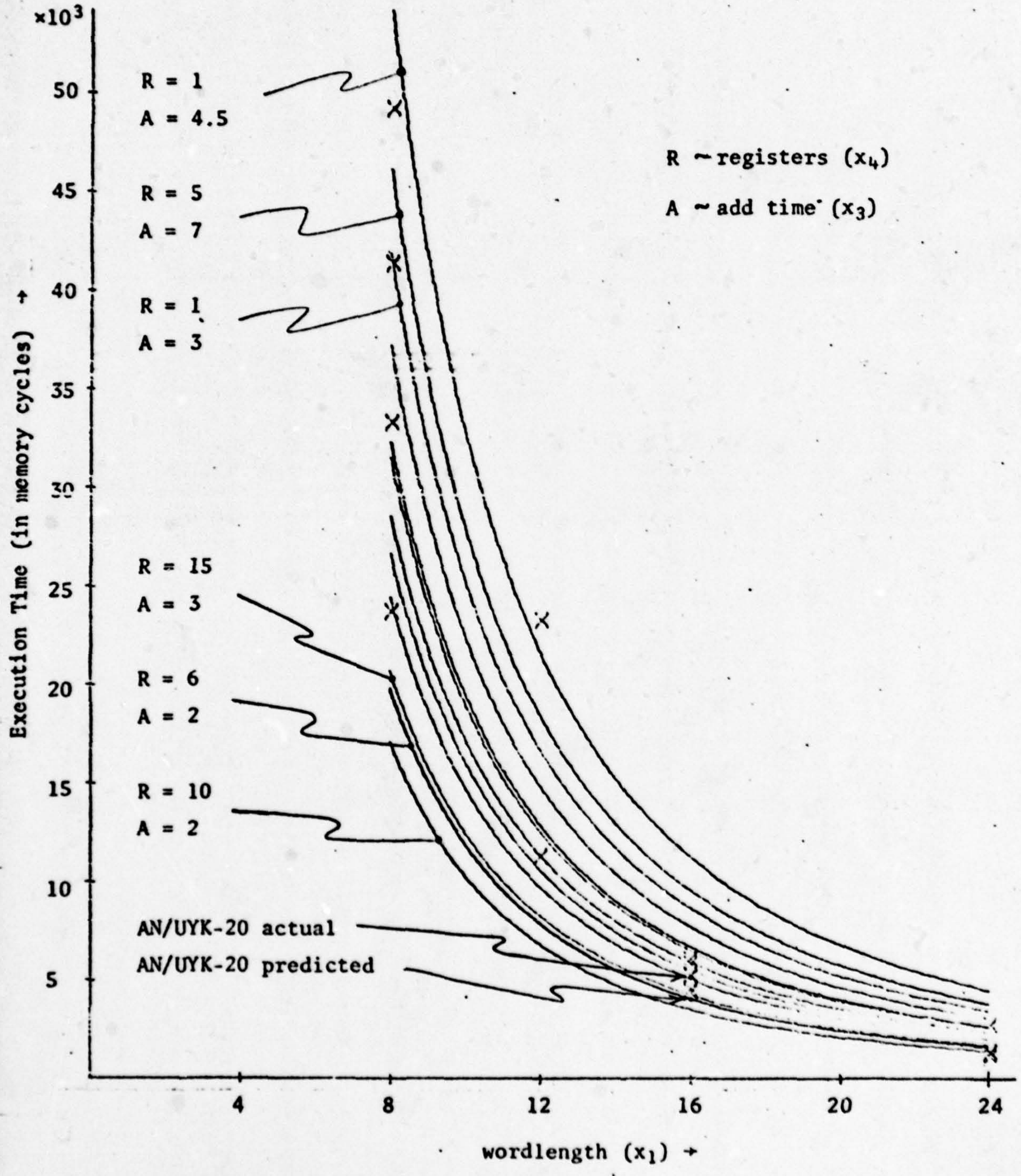


Figure 1: Regression Equation Plots for Time Performance on Kernel #1

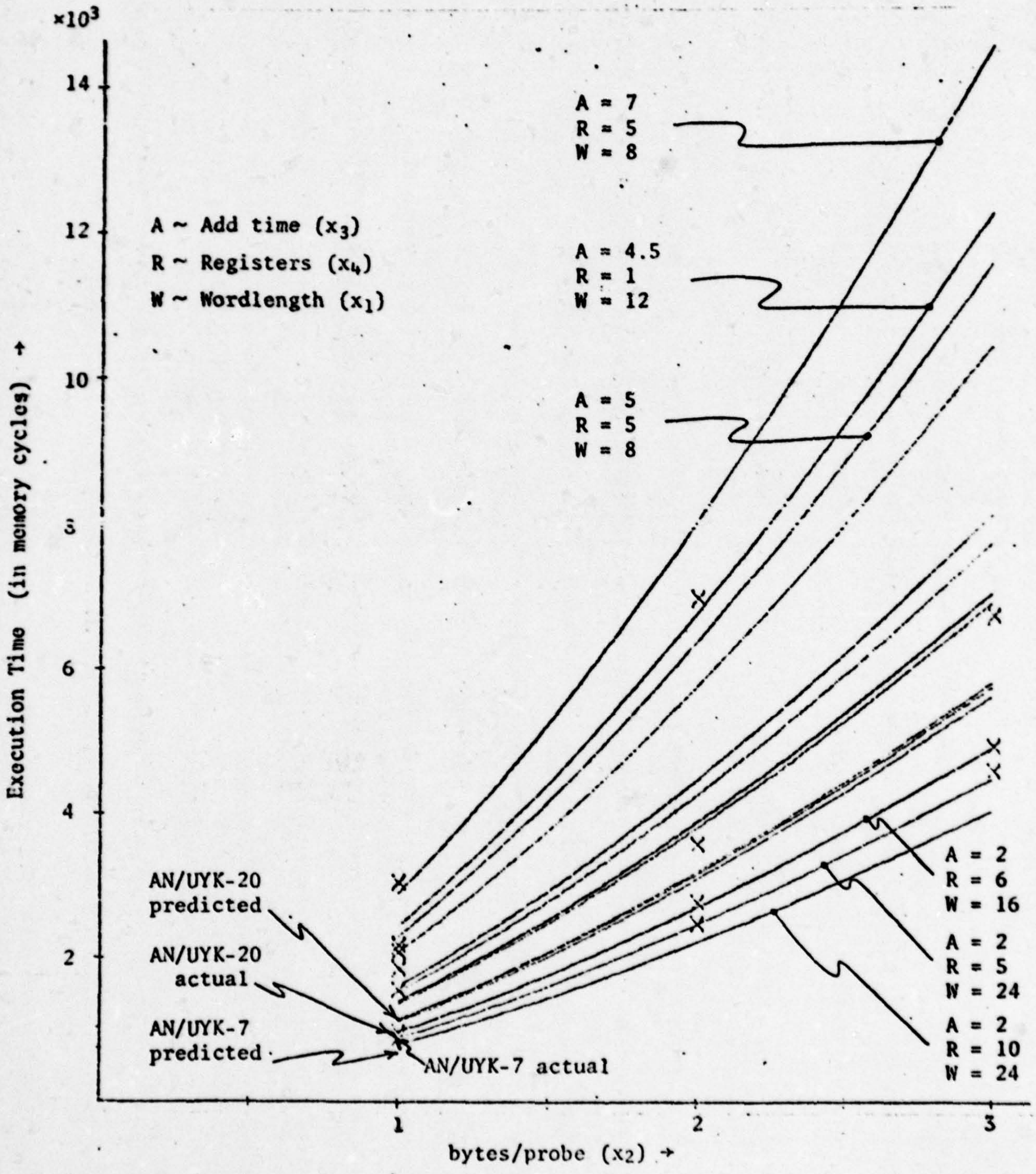


Figure 2: Regression Equation Plots for Time Performance on Kernel #2

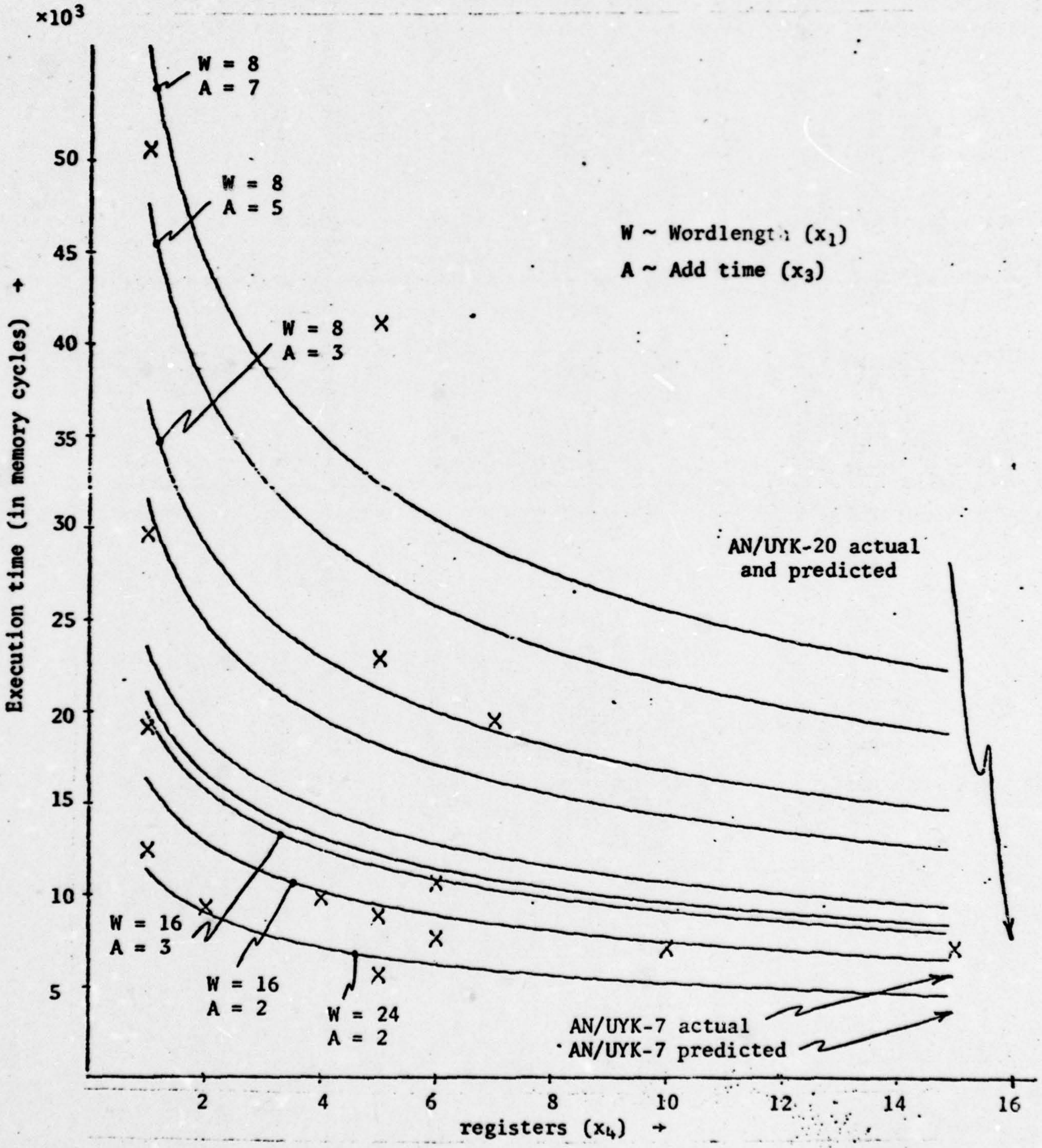


Figure 3: Regression Equation Plots for Time Performance on Kernel #3

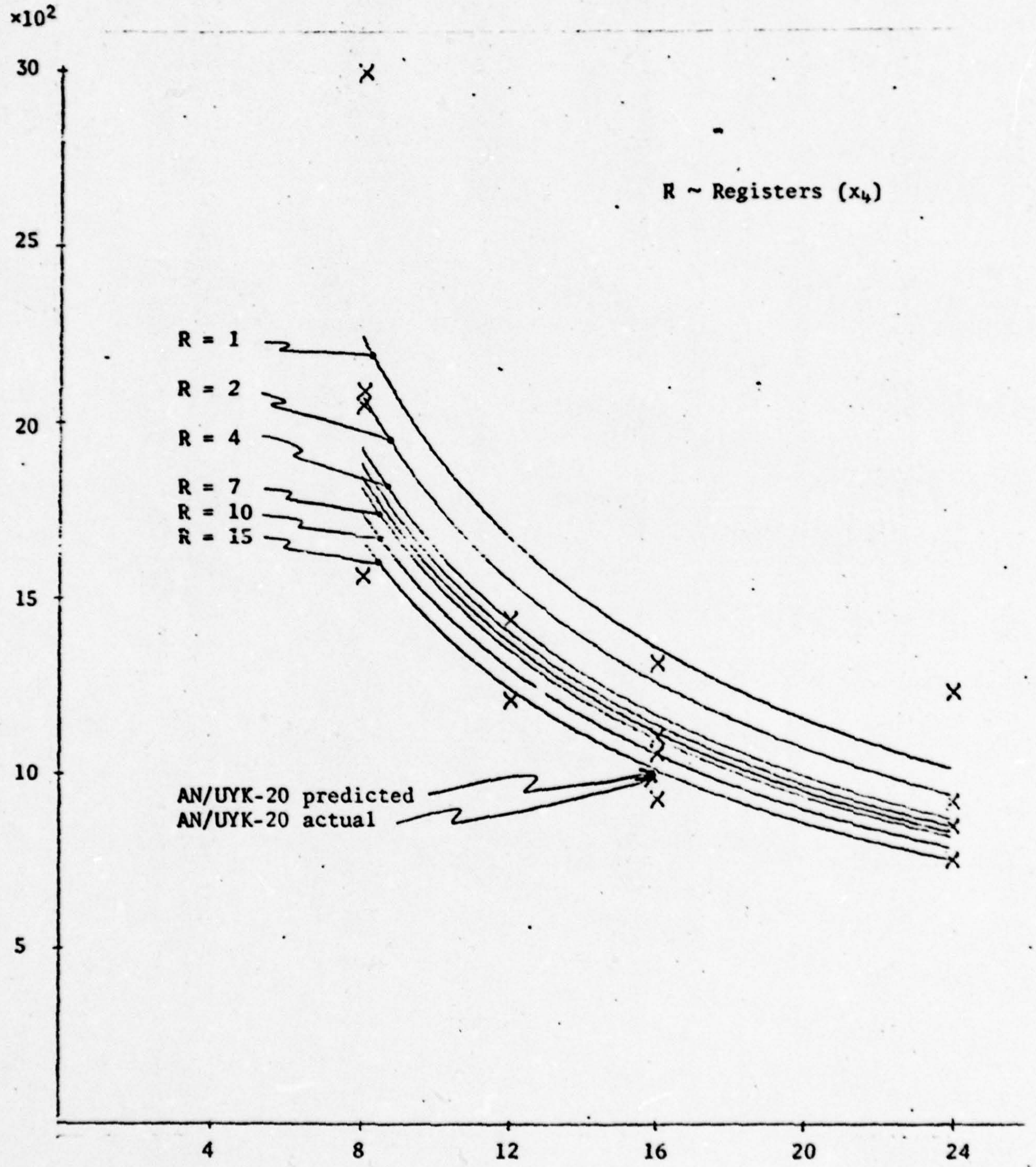


Figure 4: Regression Equation Plots for Space Performance on Kernel #1

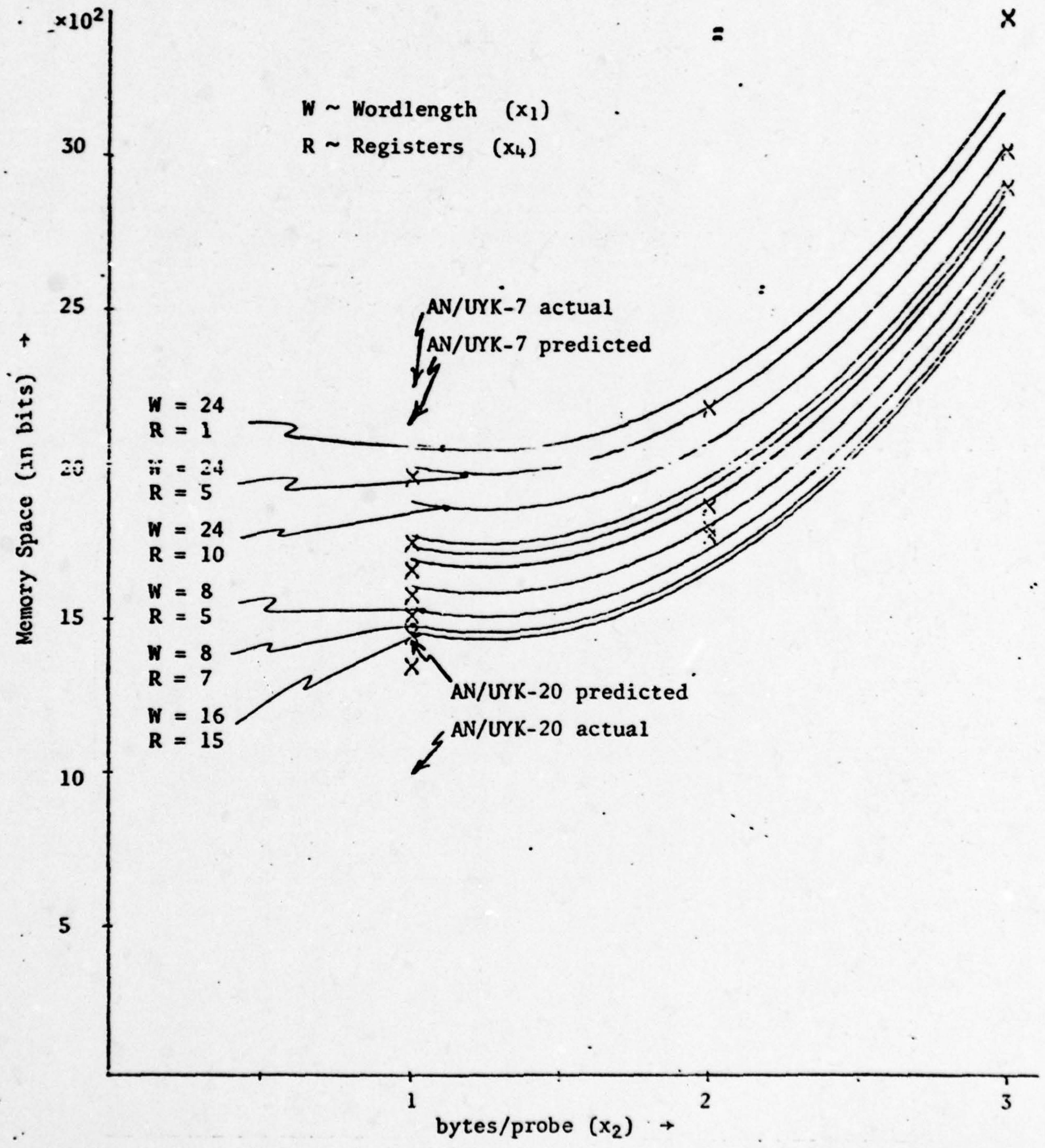


Figure 5: Regression Equation Plots for Space Performance on Kernel #2

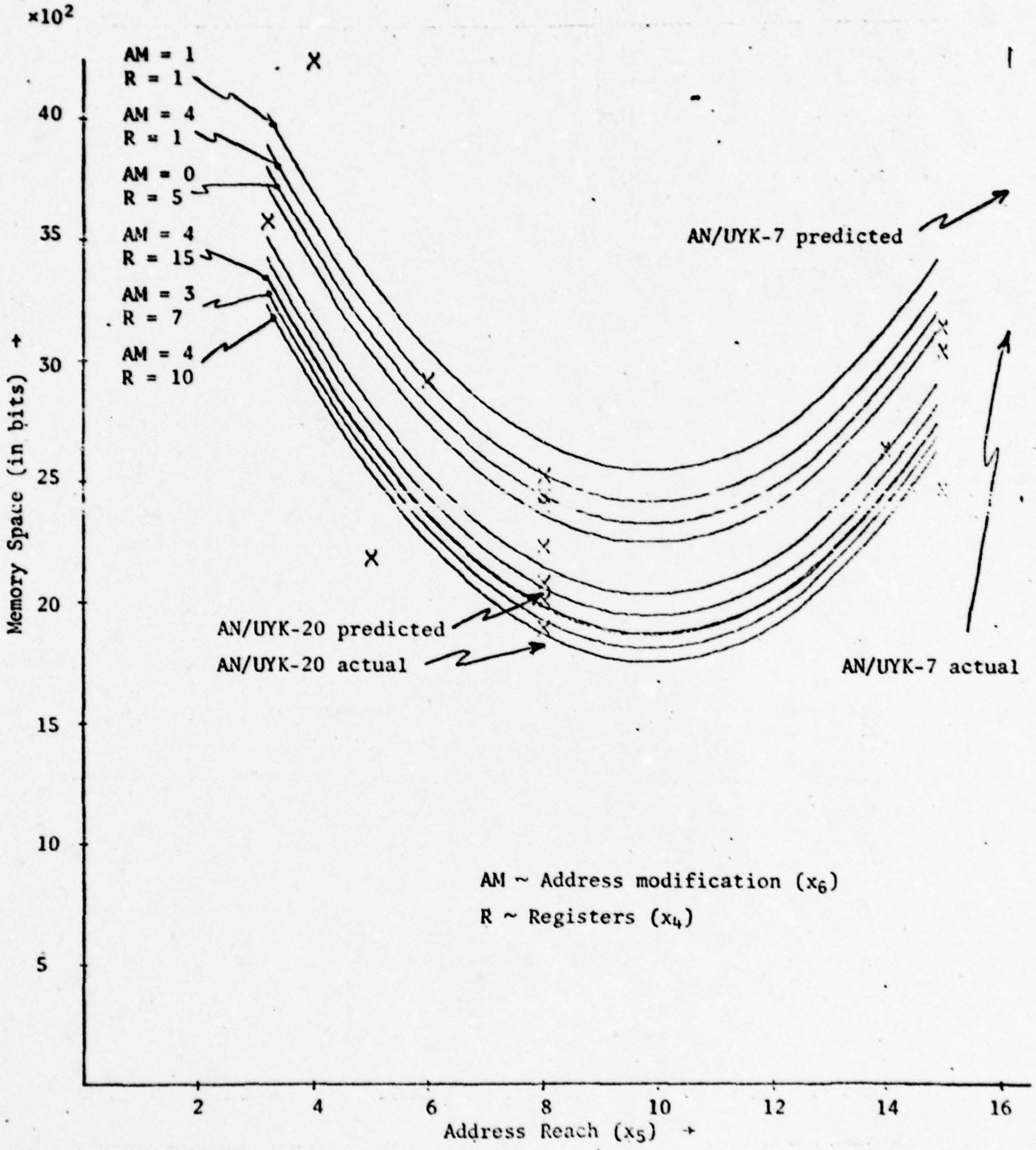


Figure 6: Regression Equation Plots for Space Performance on Kernel #3

4. Discussion, Use and Limitation of Results

In this section, we will first discuss some specific conclusions that can be drawn regarding the AN/UYK computers. This discussion will include a listing of the changes that result in the regression equations when the AN/UYK-20 is included with the original fifteen computers. The general use of the results, in particular the use of the six Figures of Section 3, will then be sketched. Finally, some limitations of the statistical technique itself will be mentioned.

4.1 Conclusions Regarding the AN/UYK-20 and AN/UYK-7 Computers

We begin with a summary evaluation of the AN/UYK-20 in each benchmark area, as compared to the other five 16-bit wordlength machines:

(a) The AN/UYK-20 on Kernel #1 (high-precision arithmetic)

The range of execution times for the six 16-bit wordlength machines on this kernel is 4616 → 6296 (see Table 2). The AN/UYK-20 is third best at 5202, slightly better than the average time of 5409. We should note that only ADD and SUBTRACT arithmetic instructions were used in this benchmark in performing division. Therefore, whether or not a machine has multiply and/or divide instructions has no bearing on the results. The space range among 16-bit machines is 944 → 1328, with an average of 1088. The AN/UYK-20 is thus also above average on this measure, ranking second at 992.

(b) The AN/UYK-20 on Kernel #2 (character manipulation)

As can be guessed, the machines that have byte addressing capability perform best here. The time and space ranges for 16-bit wordlength machines are 922 → 2818 and 1040 → 2192, with averages of 2165 and 1680, respectively.

The AN/UYK-20 is best among the 16-bit computers on both performance measures on this kernel. The time performance was reasonably predicted by the regression equations (see Tables 6 and 7) but the AN/UYK-20 is anomolously good in space performance. Our prediction was high by 37%, whereas the average prediction error was 6% and the range was 0 → 13% over all fifteen computers in the original study. A possible reason for the relatively good performance of the AN/UYK-20 on this kernel, that is not accounted for by our equation variables, is the existence of a small (4-bit) immediate operand field in one of the instruction formats. Use of this instruction type has lead to increased coding space efficiency in many parts of this kernel.

(c) The AN/UYK-20 on Kernel #3 (list processing)

The time and space ranges for the 16-bit wordlength computers are 7,256 → 10,796 and 1,808 → 2,416, with averages of 8813 and 2079, respectively. The AN/UYK-20 is again well above average, ranking second on time (7,580) and first on space (1,808).

An overall ranking of the six 16-bit wordlength computers derived by summing the time and space values for each computer over all kernels is:

	Total Time, T (over 3 kernels)		Total Space, S (over 3 kernels)	
Modcomp III	12,884	(1)	4,544	(2)
AN/UYK-20	13,704	(2)	3,840	(1)
Kongsberg 400	15,129	(3)	4,960	(4)
DataMate 16	18,269	(4)	5,440	(5)
Supernova	18,278	(5)	5,680	(6)
PDP-11/20	18,464	(6)	4,616	(3)

If we take the Time × Space product, the ranking becomes:

	T × S (10^7)
AN/UYK-20	5.262
Modcomp III	5.854
Kongsberg 400	7.504
PDP-11/20	8.523
DataMate 16	9.938
Supernova	10.380

Since the AN/UYK-7 is the only 32-bit wordlength computer in the study, the only point of interest is how well the regression equations actually predicted its performance. The equations cannot be expected to be very accurate when applied outside of the range of parameters for which they were derived. However, the prediction errors for the AN/UYK-7 are not unreasonable when compared with the range of prediction errors over the original fifteen computers (see Table 7). We should note that the execution times for the AN/UYK-7 do not account for the fact that overlapped memory bank accessing is possible in this machine, since the program can be stored in one memory bank, and the data in another. Examination of the two kernels on which the AN/UYK-7 was evaluated indicate that execution times can be reduced by an average of about 23% if instruction and data fetching are overlapped.

Finally, as an indication of the sensitivity of the regression equation constants to a change in the sample data points, we have added the data from the AN/UYK-20 to the data from the original fifteen computers and recomputed the equations. The new equations, along with the constants from the old ones (in parentheses) are:

$$T\#1 = \overset{(2.29)}{2.25 \times 10^{6}}(x_1^{\overset{(-2.26)}{-2.27}})(x_4^{\overset{(-0.276)}{-0.250}})(x_3^{\overset{(0.640)}{0.638}})$$

$$S\#1 = \overset{(1.01)}{1.01 \times 10^{4}}(x_1^{\overset{(-0.720)}{-0.720}})(x_4^{\overset{(-0.109)}{-0.111}})$$

$$T\#2 = \overset{(1780)}{1830}(x_2^{\overset{(1.48)}{1.50}})(x_3^{\overset{(0.687)}{0.693}})(x_4^{\overset{(-0.155)}{-0.168}})(x_1^{\overset{(-0.290)}{-0.304}})$$

$$S\#2 = \overset{(2130)}{2110} - \overset{(-964)}{1020x_2} + \overset{(385)}{403x_2^2} + \overset{(2.97)}{9.08x_1} + \overset{(0.807)}{0.587x_1^2} - \overset{(-18.7)}{0.0064x} - \overset{(-0.296)}{2.26x_4^2}$$

$$T\#3 = \overset{(1.35)}{1.35 \times 10^{5}}(x_4^{\overset{(-0.343)}{-0.345}})(x_1^{\overset{(-0.885)}{-0.885}})(x_3^{\overset{(0.497)}{0.497}})$$

$$S\#3 = \overset{(6140)}{6110} - \overset{(-658)}{654x_5} + \overset{(33.3)}{33.1x_5^2} - \overset{(-149)}{268x_4} + \overset{(6.83)}{46.9x_4^2} - \overset{(-252)}{138x_6} + \overset{(41.7)}{5.60x_6^2}$$

4.2 General Conclusions

The overall tendency of the performance curves given in Figures 1 - 6 suggests the following:

1. In the range of applications represented by the benchmark programs in this study, the optimum wordlength is around 16 bits. Longer wordlengths cannot be used efficiently, particularly in character-oriented applications. Shorter wordlength results in a limited address reach. This leads to an increase in addressing overhead because of the use of such techniques as

paging, indirection, etc. Figures 1 and 4 illustrate the effect of wordlength on performance most clearly, and Figure 6 shows an "optimum" space result when address reach is around 8 - 10. It should be noted that address reach is strongly correlated with wordlength. The range 8 - 10 in address reach corresponds to a wordlength range of 16 to 18 bits.

2. The number of general purpose registers has a pronounced effect on performance, as can be seen in Figure 3. However, increasing the number of registers beyond about 6 or 8 seems to have very little effect. Possibly, the programmer cannot make effective use of a larger number. While it may be argued that the threshold is programmer- and program-dependent, we do not believe that a substantial change in performance can be achieved by increasing the number of registers beyond 6 or 8.

In what follows, we present a possible general interpretation for the results of this study. The design parameters used as independent variables can be regarded as the "raw material" for the computer design process. The designer has to make optimum use of these parameters to maximize performance which is measured by the memory space and execution time of the benchmark programs. This interpretation is consistent with the fact that values for most of the independent variables used are likely to be chosen relatively early in the design process. These variables, for example, do not include any reference to the specifics of the instruction set of the machine, with the exception, perhaps, of the number of bits devoted to address modification, x_6. Therefore, it can be concluded that the curves obtained from the regression analysis represent fundamental tendencies dictated by these variables. On the other hand, the "scatter" of actual performance figures relative to these curves represents the variations among different designs that use the same raw material. This suggests that the distance between actual performance points and the curves is a measure of the success of the designer in optimizing the details of the instruction set. In this sense, the small amount of scatter in most of the figures given in this report indicates a rather surprising degree of uniformity in the design of commercially available computers. Alternatively, it can be regarded as an indication that this aspect of the design process is close to being optimum.

4.3 Limitations of the Statistical Technique

Three important limitations to the statistical techniques used in this report are:

1. Small sample size.
2. Adequacy of the kernels.
3. Programmer variability.

Fifteen machines were used in the original study[1] . These machines were selected rather arbitrarily. The main factor was simply the availability to the authors of adequate information. The curves derived are representative of true tendencies only to the extent that these fifteen machines are typical of the class of 8 - 24 bit computers. The authors feel, however, that 16-bit machines have been reasonably well represented.

The second important point is the choice of the benchmarks and the size of the kernels. The kernels required about 100 machine instructions on 16-bit computers. It may be argued that these kernels are not large enough to exhibit the relative merits of the machines in the environment of much larger programs. This, of course, is a general limitation of the kernel approach. Because of the fundamental nature of the parameters used in this study we feel that the general tendencies exhibited in the performance curves are not likely to change with kernel size.

In order to test programmer variability the three kernels were coded on one of the machines by a different programmer. An average variation of the order of 20% was observed in space requirements. In addition to variability among programmers, we should also recognize a second factor, namely, that a single programmer may not be equally familiar with all machines. There was an attempt made to minimize this effect in the present study by rechecking all programs after complete initial coding.

5. References

[1] S. Rannem, V.C. Hamacher, S.G. Zaky, P. Connolly, "On Relating Small Computer Performance to Design Parameters", Proceedings of the Second Annual Symposium on Computer Architecture, IEEE Publication No. 75CH0916-7C (available through the IEEE Computer Socity Publications Office, 5855 Naples Plaza, Suite 301, Long Beach, Ca. 90803), January 1975.

[2] A.H. Agajanian, "A Bibliography on System Performance Evaluation", COMPUTER, Vol. 8, No. 11, pp. 63-74, November 1975.

(This bibliography contains 440 references covering the period January 1970 to March 1975. Bibliographies covering earlier periods are listed.)

[3] W. Freiberger (ed.), Statistical Computer Performance Evaluation. Academic Press, New York and London, 1972.

APPENDIX: Flow-chart Listings

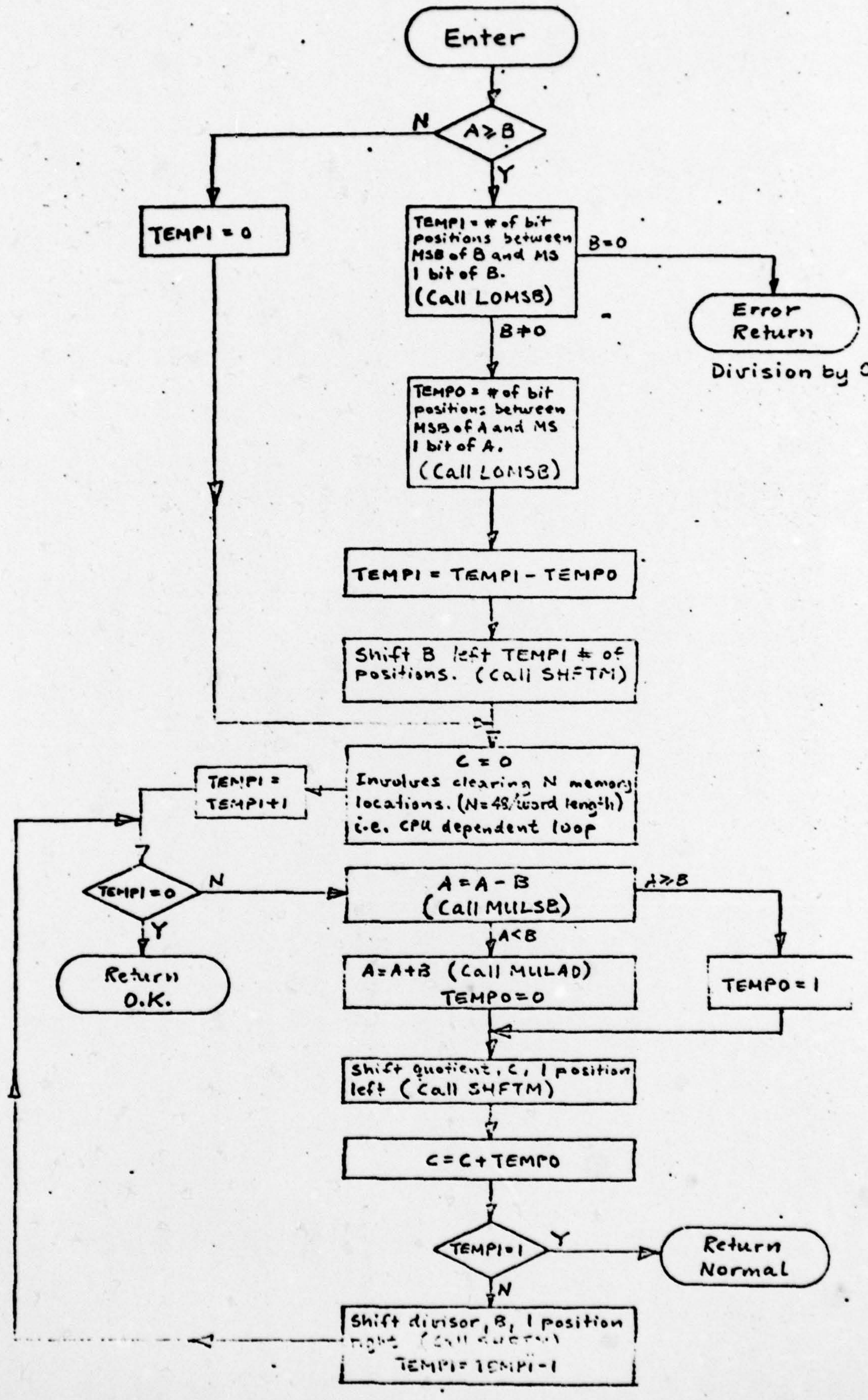
This appendix includes complete flow-chart documentation for benchmark #1, and the main program flow-charts for the other two benchmarks. The main purpose of providing these flow-charts is to give the reader a feeling for the level of complexity of the kernels. The listing includes:

The kernel for benchmark #1 consists of the routines MULAD, MULSB, and SHFTM. The kernels for the other two benchmarks are of the same level of size and complexity as that of benchmark #1.

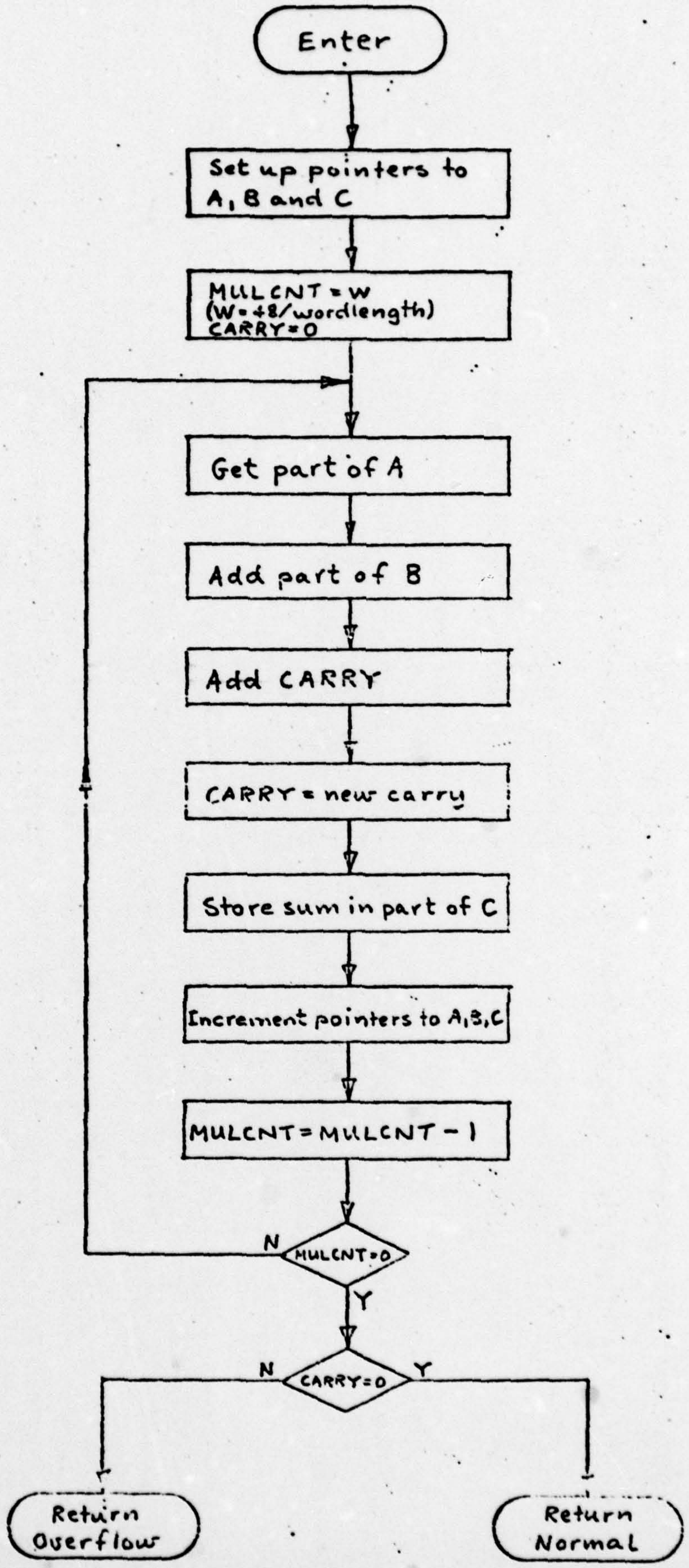
Benchmark Program No. 1

Divide

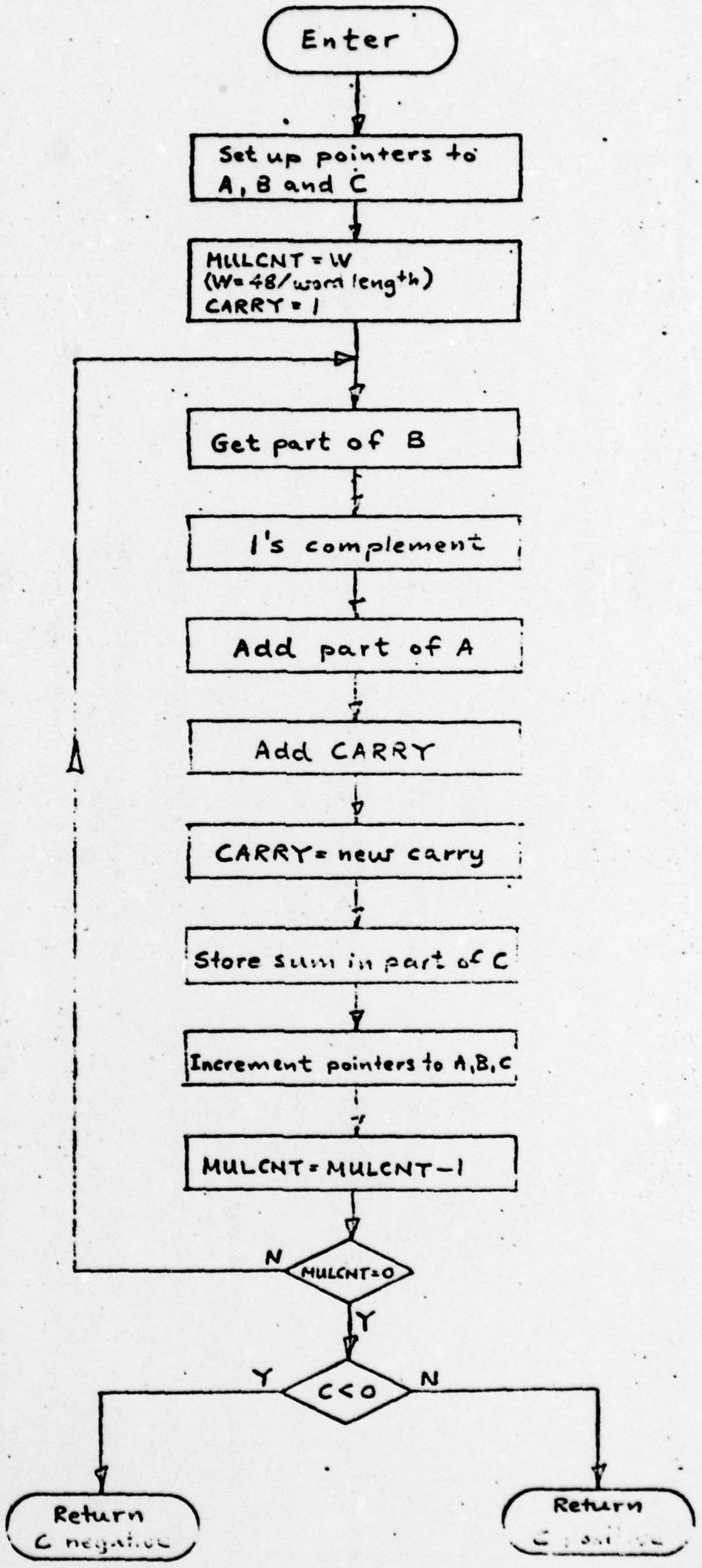
PURPOSE:	To perform division between two multiple precision (48 bits) unsigned numbers; C = A/B. (A, B and C can be anywhere in core memory.)
INPUT:	Address of dividend (A). Address of divisor (B). Address of quotient (C).
OUTPUT:	Quotient in C. Remainder in A. Divisor in B.
ROUTINES:	MULAD, MULSB, SHFTM, LOMSB.


Enter
A≥B
N
Y
TEMP1 = 0
TEMP1 = # of bit positions between MSB of B and MS 1 bit of B. (Call LOMSB)
B=0
Error Return
Division by 0
B≠0
TEMP0 = # of bit positions between MSB of A and MS 1 bit of A. (Call LOMSB)
TEMP1 = TEMP1 - TEMP0
Shift B left TEMP1 # of positions. (Call SHFTM)
C = 0
Involves clearing N memory locations. (N=48/word length) i.e. CPU dependent loop
TEMP1 = TEMP1+1
TEMP1=0
N
Y
Return O.K.
A=A-B (Call MULSB)
A≥B
A<B
A=A+B (Call MULAD) TEMP0=0
TEMP0=1
Shift quotient, C, 1 position left (Call SHFTM)
C=C+TEMP0
TEMP1=1
Y
N
Return Normal
Shift divisor, B, 1 position right (Call SHFTM)
TEMP1=TEMP1-1

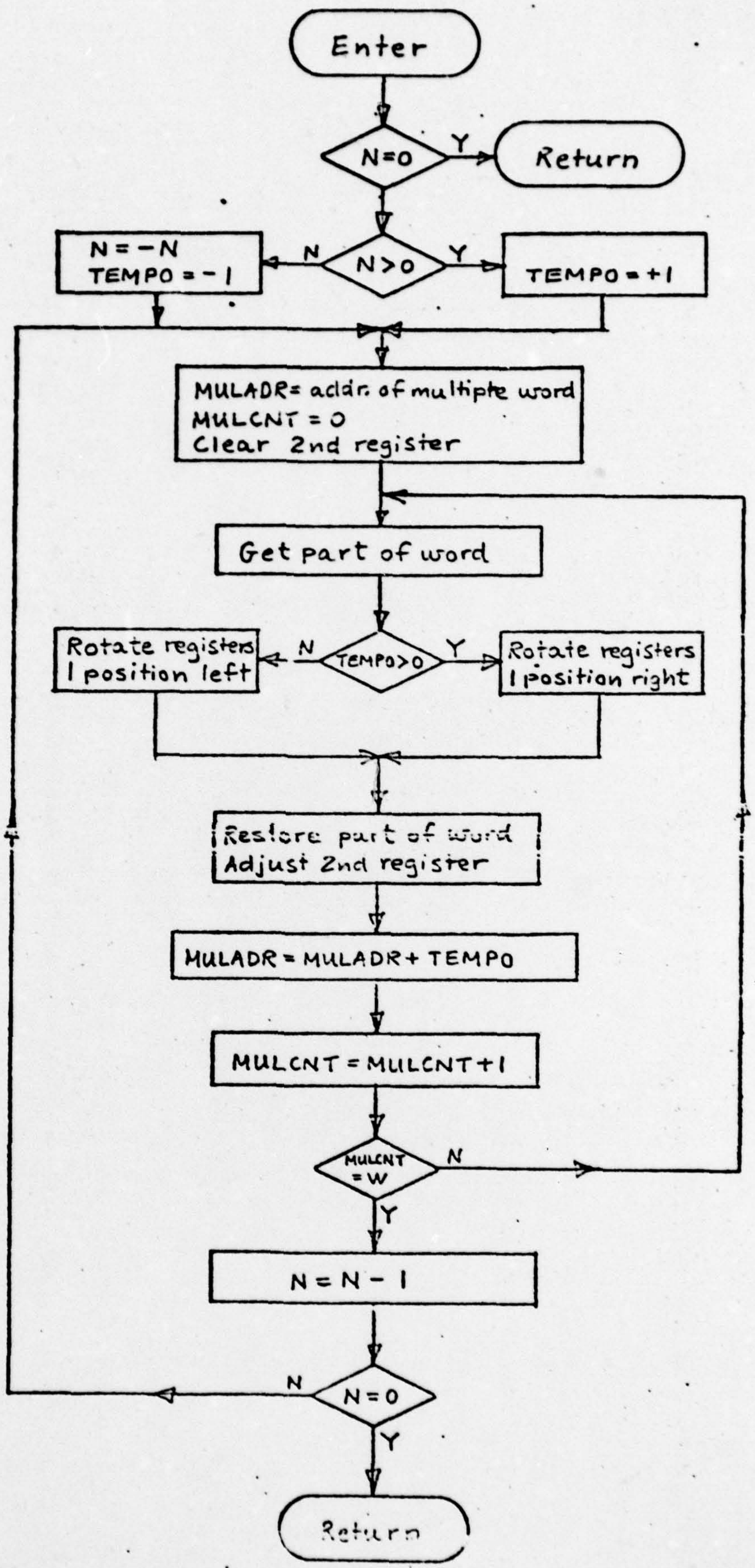
Divide


Enter
Set up pointers to A, B and C
MULCNT = W (W = 48/wordlength) CARRY = 0
Get part of A
Add part of B
Add CARRY
CARRY = new carry
Store sum in part of C
Increment pointers to A, B, C
MULCNT = MULCNT - 1
MULCNT = 0
N
Y
CARRY = 0
N
Y
Return Overflow
Return Normal

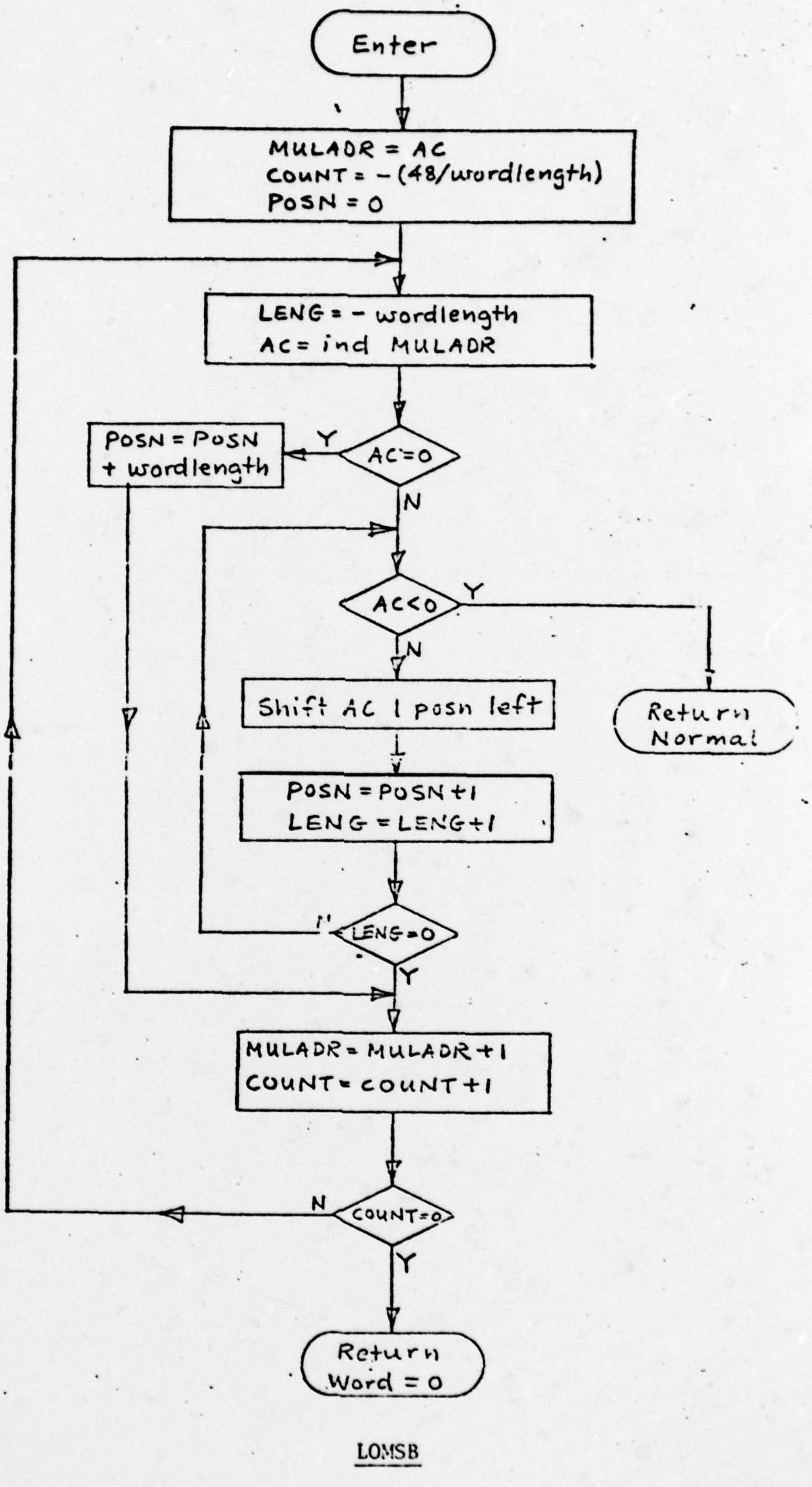
MULAD


Enter
Set up pointers to A, B and C
MULCNT = W
(W = 48/word length)
CARRY = 1
Get part of B
1's complement
Add part of A
Add CARRY
CARRY = new carry
Store sum in part of C
Increment pointers to A, B, C
MULCNT = MULCNT−1
MULCNT=0
N
Y
C<0
Y
N
Return
C negative
Return
C positive

MULSB


Enter
N=0
Y
Return
N>0
N
Y
N=-N
TEMPO=-1
TEMPO=+1
MULADR= addr. of multiple word
MULCNT=0
Clear 2nd register
Get part of word
TEMPO>0
N
Y
Rotate registers
1 position left
Rotate registers
1 position right
Restore part of word
Adjust 2nd register
MULADR=MULADR+TEMPO
MULCNT=MULCNT+1
MULCNT =W
N
Y
N=N-1
N=0
N
Y
Return

SHFTM


Enter
MULADR = AC
COUNT = -(48/wordlength)
POSN = 0
LENG = - wordlength
AC = ind MULADR
AC=0
Y
N
POSN = POSN + wordlength
AC<0
Y
N
Return Normal
Shift AC 1 posn left
POSN = POSN+1
LENG = LENG+1
LENG=0
N
Y
MULADR = MULADR+1
COUNT = COUNT+1
COUNT=0
N
Y
Return Word = 0

LOMSB

Benchmark Program No. 2

Selected Field Listing

PURPOSE: To print selected fields (actual I/O not performed) of a batch of characacter-records as specified by a format program and a list of field numbers.

Record specifications: 80 characters packed and represented in ASCII or "6-bit internal ASCII" (2 most significant bits chopped off).

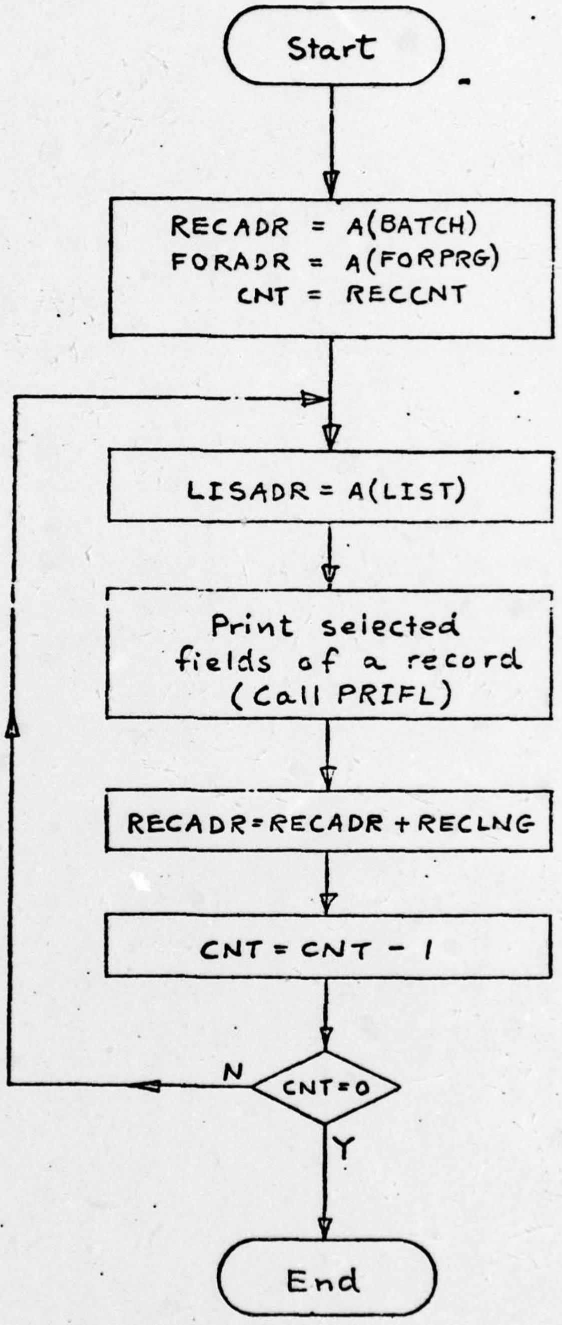
Format Program specifications: Variable length string specifying a maximum of 40 alpha and/or numeric fields. The first character of an alpha field is represented by "A". Subsequent characters are represented by "B" or "nB", where "n" is a one or two digit decimal number. The first character of a numeric field is represented by "N". Subsequent characters are represented by "M" or "nM", where "n" is as above. The string is terminated by "X".

List specifications: The list contains maximum 40 one or two digit decimal numbers. Legal numbers are 1 to 40. The list has been sorted in ascending numeric order. The first item contains the length of the list.

INPUT:

OUTPUT: The record batch is printed according to specifications.

ROUTINES: PRIFL.



Selected Field listing

Benchmark Program No. 3

Build a Balanced Binary Tree

PURPOSE: To insert information items into a binary tree and balance the tree after each insertion.

Structure of tree: The tree has a dummy node (called ROOT) serving as pointer to the root of the tree. The information in this node is a high number (usually the highest positive number that can be represented within the information area of the node). This means that when the tree is not empty, the root-pointer node (ROOT) has a valid left link and a null right link The tree is unthreaded.

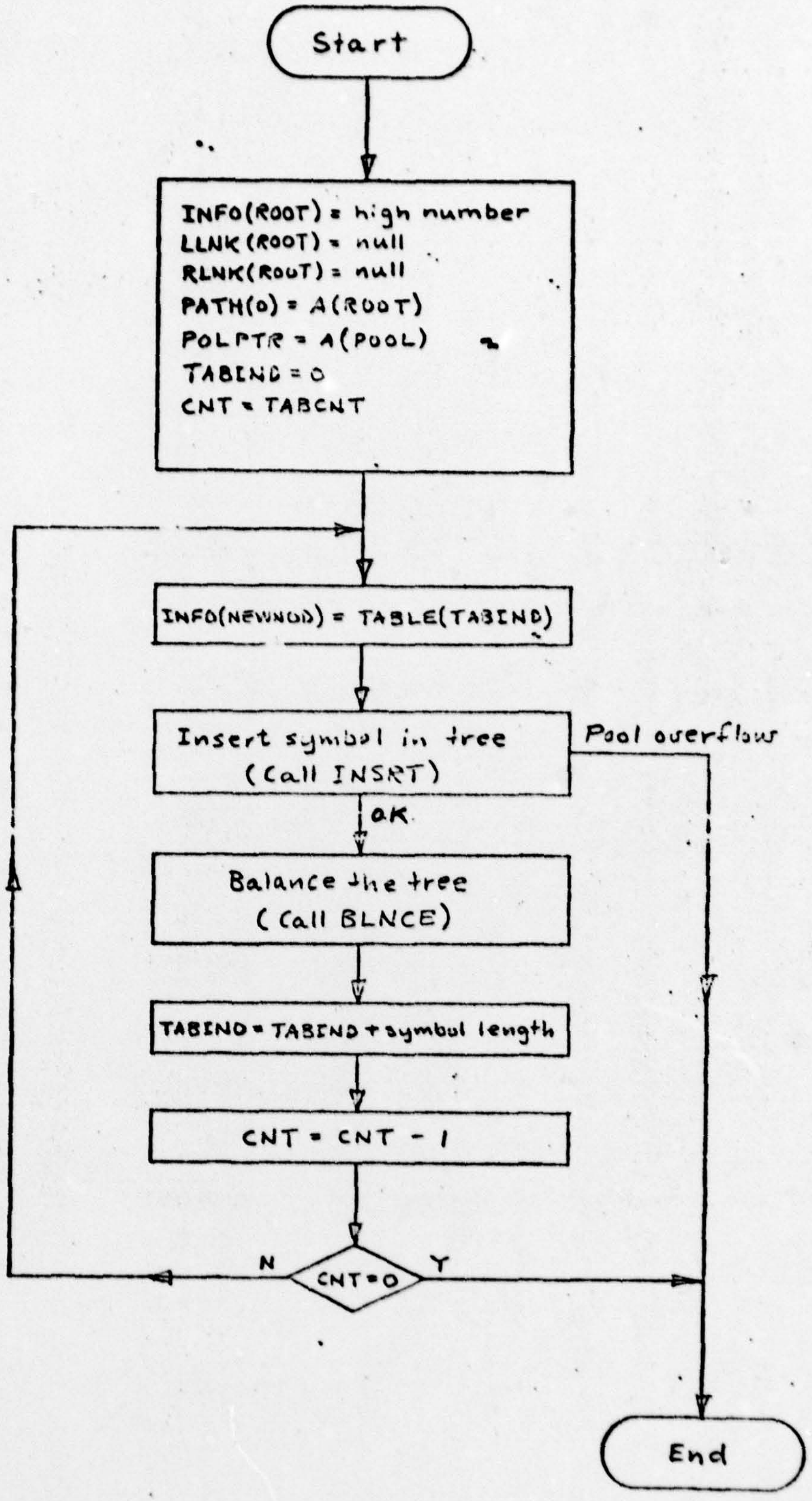
Structure of nodes:

Left link (LLNK)	Information (INFO)	Right link (RLNK)	Balance indicator (IND)
Length of address word of particular CPU.	8 bits (or more). Enough to hold an ASCII character.	Length of address word of particular CPU.	Smallest available data format ≥ 2 bits. Must be able to store and differentiate between LEFT (L), RIGHT (R) and BALANCED (B).

INPUT:

OUTPUT:

ROUTINES: INSRT, BLNCE.



Build a Balanaced Binary Tree